Contents

Install Istio with the Istio CNI plugin

7 minute read ✔ page test

Follow this guide to install, configure, and use an Istio mesh using the Istio Container Network Interface (CNI) plugin.

By default Istio injects an init container, `istio-init`, in pods deployed in the mesh. The `istio-init` container

sets up the pod network traffic redirection to/from the Istio sidecar proxy. This requires the user or service-account deploying pods to the mesh to have sufficient Kubernetes RBAC permissions to deploy containers with the `NET_ADMIN` and `NET_RAW` capabilities. Requiring Istio users to have elevated Kubernetes RBAC permissions is problematic for some organizations' security compliance. The Istio CNI plugin is a replacement for the `istio-init` container that performs the same networking functionality but without requiring Istio users to enable elevated Kubernetes RBAC permissions.

The Istio CNI plugin identifies user application pods

with sidecars requiring traffic redirection and sets this up in the Kubernetes pod lifecycle's network setup phase, thereby removing the requirement for the `NET_ADMIN` and `NET_RAW` capabilities for users deploying pods into the Istio mesh. The Istio CNI plugin replaces the functionality provided by the `istio-init` container.

Note: The Istio CNI plugin operates as a chained CNI plugin, and it is designed to be used with another CNI plugin, such as PTP or Calico. See compatibility with other CNI plugins for

details.

Install CNI

Prerequisites

1. Install Kubernetes with the container runtime supporting CNI and `kubelet` configured with the main `CNI` plugin enabled via `--network-plugin=cni`.

- AWS EKS, Azure AKS, and IBM Cloud IKS clusters have this capability.
- Google Cloud GKE clusters have CNI enabled when any of the following features are enabled: network policy, intranode visibility, workload identity, pod security policy, or dataplane v2.
- OpenShift has CNI enabled by default.

2. Install Kubernetes with the ServiceAccount admission controller enabled.
    - The Kubernetes documentation highly recommends this for all Kubernetes

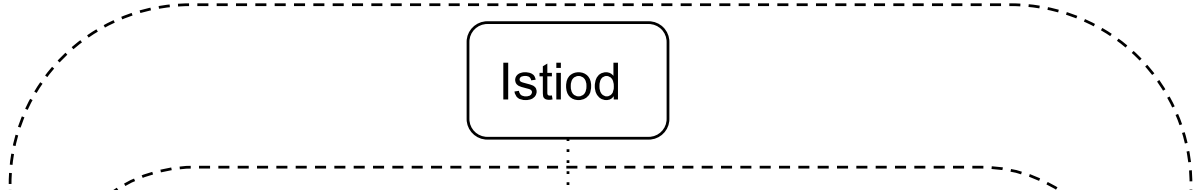
installations where `ServiceAccounts` are utilized.

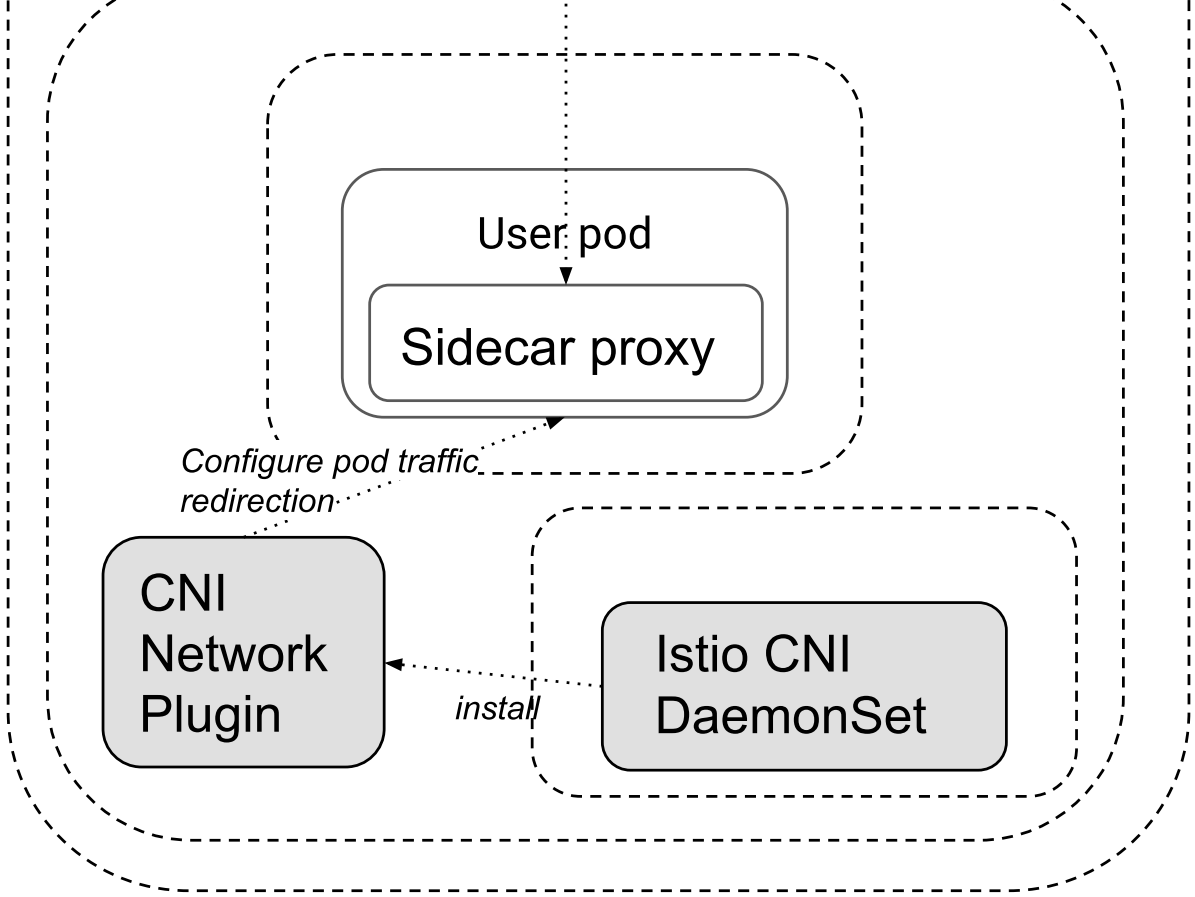
Install Istio with CNI plugin

In most environments, a basic Istio cluster with CNI enabled can be installed using the following configuration:

```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  components:
    cni:
      enabled: true
```

This will deploy an `istio-cni-node` DaemonSet into the cluster, which installs the Istio CNI plugin binary to each node and sets up the necessary configuration for the plugin. The CNI DaemonSet runs with `system-node-critical` `PriorityClass`.





Istio CNI

There are several commonly used install options:

- `components.cni.namespace=kube-system` configures the namespace to install the CNI DaemonSet.
- `values.cni.cniBinDir` and `values.cni.cniConfDir` configure the directory paths to install the plugin binary and create plugin configuration. `values.cni.cniConfFileName` configures the name of the plugin configuration file.
- `values.cni.chained` controls whether to configure the plugin as a chained CNI plugin.

There is a time gap between a node becomes schedulable and the Istio CNI plugin

becomes ready on that node. If an application pod starts up during this time, it is possible that traffic redirection is not properly set up and traffic would be able to bypass the Istio sidecar. This race condition is mitigated by a “detect and repair” method. Please take a look at race condition & mitigation section to understand the implication of this mitigation.

Hosted Kubernetes settings

The `istio-cni` plugin is expected to work with any hosted Kubernetes version using CNI plugins. The default installation configuration works with most platforms. Some platforms required special installation settings.

- Google Kubernetes Engine

```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  components:
    cni:
      enabled: true
      namespace: kube-system
  values:
    cni:
      cniBinDir: /home/kubernetes/bin
```

- Red Hat OpenShift 4.2+

```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  components:
    cni:
      enabled: true
      namespace: kube-system
  values:
    sidecarInjectorWebhook:
      injectedAnnotations:
        k8s.v1.cni.cncf.io/networks: istio-cni
    cni:
      cniBinDir: /var/lib/cni/bin
      cniConfDir: /etc/cni/multus/net.d
      cniConfFileName: istio-cni.conf
      chained: false
```

Operation details

Upgrade

When upgrading Istio with in-place upgrade, the CNI component can be upgraded together with the control plane using one `IstioOperator` resource.

When upgrading Istio with canary upgrade, because the CNI component runs as a cluster singleton, it is recommended to operate and upgrade the CNI component separately from the revisioned control

plane. The following `IstioOperator` can be used to operate the CNI component independently.

```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  profile: empty # Do not include other components
  components:
    cni:
      enabled: true
  values:
    cni:
      excludeNamespaces:
        - istio-system
        - kube-system
```

When installing revisioned control planes with the

CNI component enabled, `values.istio_cni.enabled` needs to be set, so that sidecar injector does not inject the `istio-init` init container.

```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  revision: REVISION_NAME
  ...
  values:
    istio_cni:
      enabled: true
  ...
```

The CNI plugin at version `1.x` is compatible with control plane at version `1.x-1`, `1.x`, and `1.x+1`, which

means CNI and control plane can be upgraded in any order, as long as their version difference is within one minor version.

Race condition & mitigation

The Istio CNI DaemonSet installs the CNI network plugin on every node. However, a time gap exists between when the DaemonSet pod gets scheduled onto a node, and the CNI plugin is installed and ready to be used. There is a chance that an application pod

starts up during that time gap, and the `kubelet` has no knowledge of the Istio CNI plugin. The result is that the application pod comes up without Istio traffic redirection and bypasses Istio sidecar.

To mitigate the race between an application pod and the Istio CNI DaemonSet, an `istio-validation` init container is added as part of the sidecar injection, which detects if traffic redirection is set up correctly, and blocks the pod starting up if not. The CNI DaemonSet will detect and evict any pod stuck in such state. When the new pod starts up, it should have traffic redirection set up properly. This mitigation is enabled by default and can be turned off

by setting `values.cni.repair.enabled` to false.

Traffic redirection parameters

To redirect traffic in the application pod's network namespace to/from the Istio proxy sidecar, the Istio CNI plugin configures the namespace's iptables. You can adjust traffic redirection parameters using the same pod annotations as normal, such as ports and IP ranges to be included or excluded from redirection. See resource annotations for available parameters.

Compatibility with application init containers

The Istio CNI plugin may cause networking connectivity problems for any application init containers. When using Istio CNI, `kubelet` starts an injected pod with the following steps:

1. The Istio CNI plugin sets up traffic redirection to the Istio sidecar proxy within the pod.
2. All init containers execute and complete

successfully.

3. The Istio sidecar proxy starts in the pod along with the pod's other containers.

Init containers execute before the sidecar proxy starts, which can result in traffic loss during their execution. Avoid this traffic loss with one or both of the following settings:

- Set the `traffic.sidecar.istio.io/excludeOutboundIPRanges` annotation to disable redirecting traffic to any CIDRs the init containers communicate with.

- Set the `traffic.sidecar.istio.io/excludeOutboundPorts` annotation to disable redirecting traffic to the specific outbound ports the init containers use.

Please use the above settings with caution, since the IP/port exclusion annotations not only apply to init container traffic, but also application container traffic. i.e. application traffic sent to the configured IP/port will bypass the Istio sidecar.

Compatibility with other CNI plugins

The Istio CNI plugin maintains compatibility with the same set of CNI plugins as the current `istio-init` container which requires the `NET_ADMIN` and `NET_RAW` capabilities.

The Istio CNI plugin operates as a chained CNI plugin. This means its configuration is added to the existing CNI plugins configuration as a new configuration list element. See the CNI specification reference for further details. When a pod is created or

deleted, the container runtime invokes each plugin in the list in order. The Istio CNI plugin only performs actions to setup the application pod's traffic redirection to the injected Istio proxy sidecar (using `iptables` in the pod's network namespace).

The Istio CNI plugin should not interfere with the operations of the base CNI plugin that configures the pod's networking setup, although not all CNI plugins have been validated.